AUTHORIZATION TO LEND AND REPRODUCE THE THESIS

As the sole author of this thesis, I authorize Brown University to lend it to other Institutions or individuals of the purpose of scholarly research.

_____ _____

Date Author: Yimo Zhang

I further authorize Brown University to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____ _____

Date Author: Yimo Zhang

Accounting for Missing Data in the Random Forest Algorithm

By

Yimo Zhang

Thesis

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in the Department of Biostatistics at Brown University

PROVIDENCE, RHODE ISLAND

MAY 2019

This thesis by Yimo Zhang is accepted in its present form by the Department of Biostatistics as satisfying the thesis requirement for the degree of Master of Science.

Date: 05/01/2019 Advisor: _____

Jon Steingrimsson, PhD

Reader: _____

Tao Liu, PhD

Master's Program Director: _____

Adam J. Sullivan, PhD

Approved by the Graduate Council

Date: 05/01/2019 _____

Andrew G. Campbell, PhD, Dean of the Graduate School

Acknowledgements

I would like to take this opportunity to express my gratitude to my advisor, Dr. Jon Steingrimsson, who has been patient and supportive during the last two years. He guided me through the whole research process and gave constructive suggestions on my thesis and study.

I would also like to acknowledge my thesis reader, Dr. Tao Liu, who has been giving enlightening comments on my thesis.

Finally, thank my parents for their endless love, support and encouragement that make me who I am today.

Table of contents

Table of contents

List of tables

List of figures

Chapter 1

Introduction

It is well known that failing to appropriately account for missing data can lead to bias and inefficiency [7]. Several approaches have been proposed to deal with missing data which can roughly be classified into likelihood based methods, imputation based methods, and weighing based methods [15].

Risk prediction models based on regression trees are becoming popular non-parametric alternatives to parametric regression models. The Classification and Regression Trees (CART) algorithm builds prediction models by recursively partitioning the covariate space by maximizing within group homogeneity with respect to the outcome of interest Breiman [3]. The covariate space is recursively partitioned until a pre-determined stopping criteria is met, creating a large tree (model) usually substantially overfitting the data. Pruning is used to create a sequence of candidate trees for being the final model, and cross-validation is employed to select the final tree from the candidate trees.

CART trees are easily visualized and interpretable risk prediction models. But, the hierarchical nature of the model can lead to instability as minor changes in the data can lead to different partitions high up the tree. Different splits high up the tree results in the following splits potentially being drastically different. This instability

and the simplicity of the final piece-wise constant prediction model can lead to low prediction accuracy. To improve prediction accuracy, [1] proposed bagging which averages several different regression trees built on different bootstrap samples. To further improve performance, Breiman [2] proposed the random forest algorithm. The random forest algorithm de-correlates the different trees in the bagging procedure by randomly selecting only a subset of variables to be considered for splitting.

Although extensively evaluated in the context of traditional parametric inference, little is known about how different methods for handling missing data perform when used in connection with the random forest algorithm. In this thesis we use simulations as well as analysis of data on Diabetes in Pima Indian Women to evaluate the performance of several methods for handling missing data.

1.1 Notation

Throughout the whole passage, capital letters $X, Y, Z...$ refer to random variables or fixed constants; lower case letters $x, y, z...$ refer to the realization of corresponding random variables. We introduce the following notations for the convenience of analysis.

1. N, sample size;
2. M, the number of covariates/features;
3. $\boldsymbol{Z} = (\boldsymbol{Z}_1, \boldsymbol{Z}_2, ..., \boldsymbol{Z}_N)$, the full data we intend to collect;
4. $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_N)$, the covariates/features, $\boldsymbol{X}_i = (X_i^1, X_i^2, ..., X_i^M)$;
5. $\boldsymbol{Y} = (Y_1, Y_2, ..., Y_N)$, the outcomes, and $\boldsymbol{Z}_i = (Y_i, \boldsymbol{X}_i)$;
6. $\boldsymbol{R} = (\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_N)$, and $\boldsymbol{R}_i = (R_i^0, R_i^1, ..., R_i^M)$ is the missing indicator of $\boldsymbol{Z}_i = (Y_i, X_i^1, ...X_i^M)$, where $R_i^j = 0$ indicates that the corresponding component of $(Y_i, X_i^1, X_i^2, ..., X_i^M)$ is missing (where $j = 0$ corresponds to the outcome and $j = k$ corresponds to X_i^k), $R_i^j = 1$ if the corresponding component of $(Y_i, X_i^1, X_i^2, ..., X_i^M)$ is observed;
7. $\bar{\boldsymbol{R}} = (\bar{\boldsymbol{R}}_1, ..., \bar{\boldsymbol{R}}_N)$, and $\bar{\boldsymbol{R}}_i = (\bar{R}_i^0, \bar{R}_i^1, ..., \bar{R}_i^M)$, where $\bar{R}_i^j = 1 - R_i^j$;

8. $\boldsymbol{L} = (L_1, ..., L_N)$ is the missing indicator of $\boldsymbol{Z}_i$ as a whole, where $L_i = 1$ if $\boldsymbol{Z}_i$ is fully observed, and $L_i = 0$ if $\boldsymbol{Z}_i$ is not fully observed;
9. $O = \sum_{i=1}^{n} L_i$, the number of individuals that are fully observed.

1.2 Missing Mechanism

Missingness can happen either at covariates or the outcome. The missingness type is said to be univariate if all missingness happens in a single covariate (i.e. the outcome and all other covariates are fully observed). Monotone missingness occurs if $R^j = 0$ implies $R^k = 0$ for all $k > j$. The missingness is said to be nonmonotone if it is not monotone.

Now we describe three types of missingness mechanisms:

1.2.1 Missing Completely at Random

Missing Completely at Random refers to the situations where the missing of data has nothing to do with the data itself, that is, the missingness is independent of the full data set,

$$\mathbb{P}(R = r \mid Z) \text{ does not depend on } Z.$$

The reason behind this kind of missingness often has little to do with the study itself. For example, the records were documented with random errors and the erroneous records were marked to be missing.

1.2.2 Missing at Random

Missing at Random refers to the situations where the missingness of the data depends only on the data that are observed,

$$\mathbb{P}(R = r \mid Z) = \mathbb{P}(R = r \mid Z_r) = \pi(r, Z_r),$$

where π is some function. Here, Z_r is the observed component of $\mathbf{Z}$. For example, in a non-blinded study, the patients who are not assigned to treatment group drop out, thus leave their outcome measurement unobserved.

1.2.3 Missing Not at Random

Missing not at Random refers to the situations where the probability of being missing depends on the data that are not observed,

$$\mathbb{P}(R = r \mid Z) \text{ depends on } Z_{\bar{r}}.$$

Here, $Z_{\bar{r}}$ is the missing component of $\mathbf{Z}$. This happens, for example, when patients somehow get to know in advance that the outcome is not desirable and they choose to leave the study due to negativity.

1.3 Handling Missing Data

Many methods can be adopted to address missing data problems. The most straightforward one is to completely ignore observations that are not fully observed. However, this method is not recommended in practical settings as it throws away information carried by the data set, and can lead to bias and inefficiency.

To make use of the information in partially observed observations, two popular ways of handling missing data are imputation and weighing. Now we describe the methods in general terms and in Section 2 we describe them in the context of the random forest algorithm.

Imputing the Mean

The simplest method to impute missing data is to impute the mean of observed data, that is

$$X_i^j = \frac{1 - r_i^j}{\sum_{k=1}^N r_k^j} \sum_{k=1}^N r_k^j X_k^j + r_i^j X_i^j, \quad j = 1, 2, ..., M, \tag{1.1}$$

$$Y_i = \frac{1 - r_i^0}{\sum_{k=1}^N r_k^0} \sum_{k=1}^N r_k^0 Y_k + r_i^0 Y_i. \tag{1.2}$$

However, in despite of the simplicity, this method can also lead to problematic results where variance is underestimated and the missing values are imputed using a complete case estimator. A way to allow partially missing observations to behave differently than fully observed observations is to add a missing indicator term when we fit model using this mean-imputed data. For example, in the context of a linear model

$$E[Y \mid \boldsymbol{X}_i] = \gamma_0 L_i + \sum_{j=1}^M \gamma_j X_i^j,$$

where $\boldsymbol{X}_i$'s that are conditioned on in the above statements are covariates after mean imputation.

An alternative would be to add an indicator whether a variable was missing on that participant for each variable. That would allow the missingness corresponding to each variable to be different depending on if that variable is missing or not.

Multiple Imputation

The basic idea of multiple imputation is to first impute the data based on some model a few times to obtain several full data set; then conduct analysis on each data set and calculate their own estimators, and the final estimator is the average of them and the variance estimator is adjusted for missing data.

Assume the number of imputations is K. Then the multiple imputation algorithm is given by

Algorithm 1 Algorithm of general MI

while i in 1 to K **do**

1. Impute in the missing values based on some method and create the ith data set

2. Conduct data analysis on the ith data set and attain an estimator $\hat{\theta}_i$ for some parameter θ

end while

3. The final estimator of θ is: $\hat{\theta} = \frac{1}{K}\sum_{i=1}^{K}\hat{\theta}_i$.

4. The estimator for variance of $\hat{\theta}$ is:

$$\hat{\Sigma} = \frac{1}{K}\sum_{i=1}^{K}\hat{V}ar(\theta)_i + \frac{K+1}{K(K-1)}\sum_{i=1}^{K}(\hat{\theta}_i - \hat{\theta})(\hat{\theta}_i - \hat{\theta})^T,$$

where $\hat{V}ar(\theta)_i$ is the variance estimator from the ith imputed data set.

Several methods of MI have been proposed in terms of how to impute data (fill in the missing values in **Step 1**), including proper imputation, improper imputation and multivariate imputation [11].

The method we adopt is called **Multivariate Imputation by Chain Equations** (**MICE**) [17], which imputes data again and again till convergence to create a single imputed data set. Assuming there are $B(B \leqslant M)$ variables in the data set containing missing values, then the algorithm is as follows:

Algorithm 2 MICE Algorithm for multiple imputation

1. Fill in all missing values by random sampling from the observed values.

while j in 1 to B **do**

2. Regress the jth variable with missing values, say, $\boldsymbol{X}^j$, on all other variables, using only individuals whose $\boldsymbol{X}^j$ is observed. Note that the imputed values in previous steps should also be included in the regression.

3. Impute the missing values of the jth variable using the regression model obtained in the previous step.

end while

4. Repeat the **while** loop several times until the result becomes stable.

One key advantage of this method is its compatibility to different types of responses, whether it's continuous, binary or categorical, and the use of regression model in **Step 2** (linear regression, logistic regression or multinomial regression) should depend on the type of response.

Inverse Probability Weighting

The basic idea of inverse probability weighting [10]is to model the probability of missingness and weigh the observations by the probability of being fully observed. The intuition is that for every observed individual, the less likely it is to be observed, the more likely similar individuals are supposed to be missing in the data set.

Thus, to implement inverse probability weighting, we must:

1. Fit model for probability of being observed by all variables that are fully observed. Assume that a total of J variables, $\{\boldsymbol{W}^1, \boldsymbol{W}^2, ..., \boldsymbol{W}^J\}$, are fully observed, where $\{\boldsymbol{W}^1, \boldsymbol{W}^2, ..., \boldsymbol{W}^J\} \subseteq \{\boldsymbol{X}^1, \boldsymbol{X}^2, ..., \boldsymbol{X}^M, \boldsymbol{Y}\}$. Fit a logistic regression model for

the probability of $L = 1$

$$P(L = 1|\mathbf{X}, Y) = \text{logit}\Big[\pi(\boldsymbol{Z_i}; \boldsymbol{\beta})\Big] = \beta_0 + \sum_{k=1}^{J} \beta_k W_i^k. \tag{1.3}$$

Other prediction models can also be used.

2. Using the model fitted above to estimate the parameter of interest by solving a weighted estimating equation (or a weighted version of the analysis of interest).

Chapter 2

Methods

2.1 Random Forest

2.1.1 Building a Single Tree

The regression and classification tree algorithm is supervised learning algorithm. To build a tree, we partition the feature space into small regions, and then fit a simple model for each region. In figure 2.1, a tree is built by partitioning a two-dimensional space into five regions, R_1, R_2, R_3, R_4, R_5 using the splits $X_1 = t_1, X_1 = t_3, X_2 = t_2, X_2 = t_4$. Once the tree is built, we can make prediction on a new feature vector $\boldsymbol{x} = (x^1, x^2)$ by

$$T(\boldsymbol{x}) = \sum_{i=1}^{5} \hat{f}_i(\boldsymbol{x}) I_{\{\boldsymbol{x} \in R_i\}}, \tag{2.1}$$

where

$$I_{\{\boldsymbol{x} \in R_i\}} = \begin{cases} 1 & \text{if } \boldsymbol{x} \in R_i \\ 0 & \text{if } \boldsymbol{x} \notin R_i \end{cases}$$

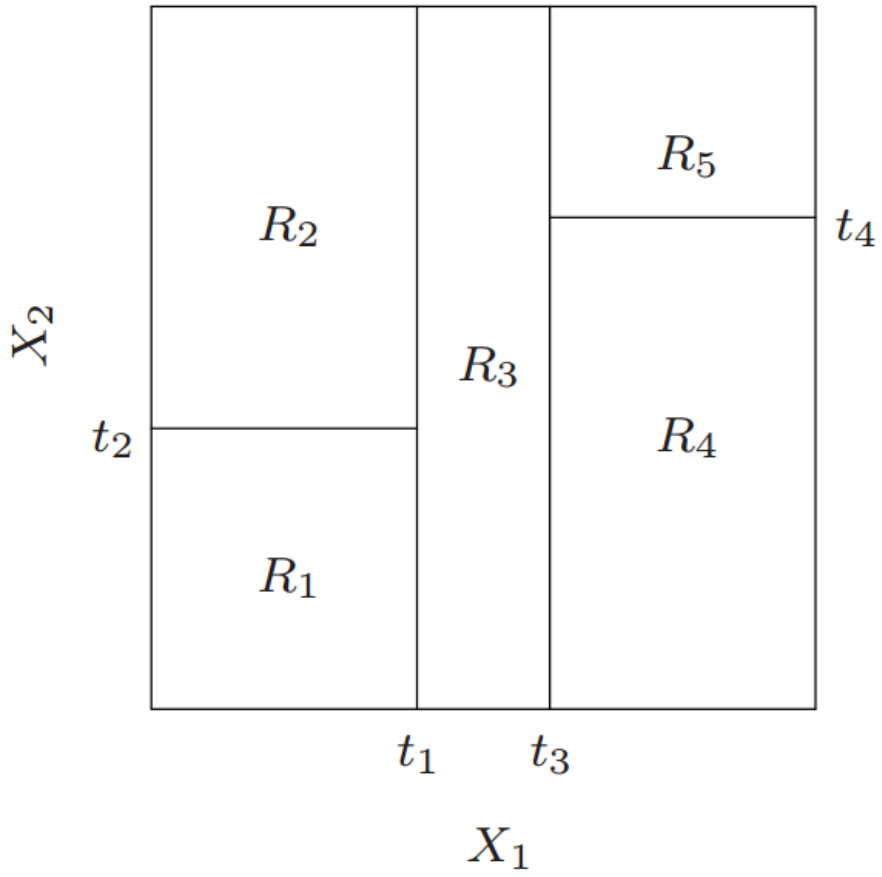

Fig. 2.1 Tree Built by Partitioning a Two-Dimensional Feature Space (adopted from [14])

is the indicator function and $\hat{f}_i(\boldsymbol{x})$ is a specific model fitted using data from region R_i. When the outcome is continuous and a squared error loss is used, $\hat{f}_i$ is a constant by averaging all the outcomes whose features fall in region R_i, that is,

$$\hat{f}_i = \hat{C}_i = \frac{1}{\sum_{j=1}^{N} I_{\{\boldsymbol{X}_j \in R_i\}}} \sum_{j=1}^{N} I_{\{\boldsymbol{X}_j \in R_i\}} Y_j. \tag{2.2}$$

where the model $\hat{f}_i = \hat{C}_i$ minimizes mean squared error in each region:

$$\hat{C}_i = \min_{C_i \in \mathcal{R}} \sum_{j=1}^{N} I_{\{\boldsymbol{X}_j \in R_i\}} (Y_j - C_i)^2.$$

After determining the regression model in each region, we then start to grow the tree by deciding:

1. Which feature/variable and nodes to split;
2. When to stop.

The first step is solved by first going over all possible splitting variables and nodes, and selecting the pair of variable and node (j, s) that minimizes the total sum of squares,

$$\min_{j,s} \Big[\sum_{k:\boldsymbol{x}_k \in R_1} (y_k - \hat{f}_1)^2 + \sum_{k:\boldsymbol{x}_k \in R_2} (y_k - \hat{f}_2)^2 \Big], \tag{2.3}$$

where $\hat{f}_i$, $i = 1, 2$ is calculated by (1) with regions $R_1 = \{\boldsymbol{X}_k : X_k^j < s\}$ and $R_2 = \{\boldsymbol{X}_k : X_k^j \geqslant s\}$. After making the first split, continue this process for every resulting region. The second steps is achieved when some pre-specified stopping criteria is met (e.g. the depth of the tree is large enough or there is not enough data in each node to warrant further splitting).

2.1.2 Bagging

After building a tree T, a prediction can be made for a new individual x,

$$T(\boldsymbol{x}) = \sum_{k=1}^{|T|} \hat{f}_k I_{\{\boldsymbol{x} \in R_k\}},$$

where $|T|$ is the number of regions of tree T. However, basing the prediction on only one tree can be unreliable as the splitting criteria can easily be affected by outliers. Bagging is a technique for reducing the variance of prediction by averaging the predictions yielded by models built from bootstrap samples. It's argued that bagging helps reduce mean-squared error and leaves the bias unchanged [1].

Assume we draw a total of B bootstrap samples from $(\boldsymbol{Z}_1, \boldsymbol{Z}_2, ..., \boldsymbol{Z}_N)$ and the tree built from the bth bootstrap sample is marked as T^{*b}, then the final prediction is

$$T_{bag}(\boldsymbol{x}) = \frac{1}{B} \sum_{b=1}^{B} T^{*b}(\boldsymbol{x}).$$

2.1.3 Random Forest

Bagging method improves prediction accuracy produced by a single tree. However, the variance is also considerable since each tree built from bootstrap sample has large correlation since they use the same splitting criteria and almost the same data. Assume the B bootstrap trees are identically distributed with variance σ^2 and pairwise correlation ρ, then the variance of the bagging average (final prediction) is:

$$\text{var}\Big[T_{bag}(\boldsymbol{x})\Big] = \text{var}\Big[\frac{1}{B} \sum_{b=1}^{B} T^{*b}(\boldsymbol{x})\Big] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2,$$

and

$$\lim_{B\to\infty} \text{var}\Big[T_{bag}(\boldsymbol{x})\Big] = \rho\sigma^2.$$

The idea of random forest is to reduce the correlation between trees ρ without increasing the variance σ^2 too much. To achieve this, instead of applying splitting criteria to all features on all bootstrap trees, we **randomly** choose P features out of M as the candidates for each splitting. In this way, even though the splitting criteria is the same, each bootstrap tree will likely look much different.

Algorithm 3 Random Forest Algorithm for Fully Observed Data

while i in 1 to B **do**

1. Draw the bth bootstrap sample $\mathbf{Z}^{*b}$ of size N from $\mathbf{Z}$

2. Build the bth tree T^{*b} by (2.1) on the bootstrapped sample from previous step by repeating the following steps on each node until the number of nodes of T^{*b} reaches some threshold.

i. Select P features at random from the M features.

ii. Find the best split pair of feature and node (j, s) by solving (2.3) and split that node.

end while

3. The final prediction $T_{rf}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} T^{*b}(\boldsymbol{x})$

2.2 Handling missing data in the Random Forest algorithm

When using random forest to train an incomplete data set, we first address the missing data problem by either imputing or inverse probability weighting to obtain a complete data set, and then implement random forest algorithm on this data set.

2.2.1 Multiple Imputation by Chain Equation

For a data set yielded by imputing, we can just treat it as an original data set and simply run a random forest algorithm.

Algorithm 4 Random Forest Algorithm with MICE

while m in 1 to K **do**

1. Impute the data set using algorithm 2 to obtain the mth data set $Z_{(m)}$.

while i in 1 to B **do**

2. Draw the bth bootstrap sample $\mathbf{Z}^{*b}_{(m)}$ of size N from $\mathbf{Z}_{(m)}$

3. Build the bth tree $T^{*b}_{(m)}$ by (2.1) on the bootstrapped sample from previous step by repeating the following steps on each node until the number of nodes of $T^{*b}_{(m)}$ reaches some threshold.

i. Select P features at random from the M features.

ii. Find the best split pair of feature and node (j, s) by solving (2.3) and split that node.

end while

4. The prediction based on the mth imputed data set is: $T^{(m)}_{rf}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} T^{*b}_{(m)}(\boldsymbol{x})$.

end while

5. The final prediction is the average based on the K imputed data sets: $T_{rf}(\boldsymbol{x}) = \frac{1}{K}\sum_{m=1}^{K} T^{(m)}_{rf}(\boldsymbol{x})$.

2.2.2 Inverse Probability Weighting

To incorporate inverse probability weighting into the random forest algorithm, some modifications need to be made.

There are two ways to incorporate weights in random forest algorithm. One is to use the weighted bootstrap instead of the regular bootstrap Præstgaard and Wellner [9]:

Algorithm 5 IPW Random Forest Algorithm

1.Estimate the weights for each observed individual $\hat{\omega}_i = \frac{1}{\hat{\pi}_i}$ using (1.3). When observations are not fully observed $L = 0$ the weights are 0 so observations that are not fully observed are not selected in the bootstrap sample.

while i in 1 to B **do**

2. Draw the bth weighted bootstrap sample $\mathbf{Z}^{*b}$ of size O with probability $\frac{\hat{\omega}_i}{\sum_{i:L_i=1}\hat{\omega}_i}$ from $\{\boldsymbol{Z}_i : L_i = 1\}$

3. Build the bth tree T^{*b} by (2.1) on the bootstrapped sample from previous step by repeating the following steps on each node until the number of nodes of T^{*b} reaches some threshold.

i. Select P features at random from the M features.

ii. Find the best split pair of feature and node (j, s) by solving (2) and split that node.

end while

4. The final prediction $T_{rf}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} T^{*b}(\boldsymbol{x})$

By only putting positive weights on fully observed observations in the bootstrap sample, Algorithm 5 ensures that the individual trees in the bootstrap sample are built only on fully observed data.

The other way to use inverse probability weighting to account for missing data in the random forest algorithm is to apply a different splitting criteria:

Algorithm 6 Random Forest Algorithm using IPW based Splitting Criteria

1. Estimate the weights for each observed individual $\hat{\omega}_i = \frac{1}{\hat{\pi}_i}$ using 1.3

while i in 1 to B **do**

2. Draw the bth bootstrap sample $\mathbf{Z}^{*b}$ of size N from fully observed observations in $\boldsymbol{Z}$.

3. Build the bth tree T^{*b} by (2.1) on the bootstrapped sample from previous step by repeating the following steps on each node until the number of nodes of T^{*b} reaches some threshold.

i. Select P features at random from the M features.

ii. Find the best split pair of feature and node (j, s) by solving the following and split that node.

$$\min_{j,s} \Big[\sum_{k:\boldsymbol{x}_k \in R_1} \hat{\omega}_k (y_k - \hat{f}_1)^2 + \sum_{k:\boldsymbol{x}_k \in R_2} \hat{\omega}_k (y_k - \hat{f}_2)^2 \Big], \tag{2.4}$$

where $\hat{f}_i = \frac{1}{\sum_{k:\boldsymbol{x}_k \in R_i} \hat{\omega}_k} \sum_{k:\boldsymbol{x}_k \in R_i} \hat{\omega}_k Y_k$ with regions $R_1 = \{\boldsymbol{X}_k : X_k^j < s\}$ and $R_2 = \{\boldsymbol{X}_k : X_k^j \geqslant s\}$.

end while

4. The final prediction $T_{rf}(\boldsymbol{x}) = \frac{1}{B} \sum_{b=1}^{B} T^{*b}(\boldsymbol{x})$

Chapter 3

Simulation

3.1 Set Up

We conducted 100 simulations to evaluate the performance of random forest algorithm when incorporating the methods to address missing data discussed in the previous section. We used simulation settings that included both continuous and binary outcomes. In each simulation, a data set of size 1000 was created, with 15 variables, $X_1, X_2, ..., X_{15}$ generated from multivariate normal distribution with mean $\mathbf{0}$. The covariance of any pair of covariates (X_i, X_j) was set to be $0.9^{|i-j|}$. The continuous outcome was then calculated using five covariates through the following formula:

$$Y_{con} = 5 + 4X_1 + 3e^{X_2} + 3X_3^3 - X_{11}^4 + 2X_{12}^3 + \varepsilon \tag{3.1}$$

where $\varepsilon \sim N(0,1)$ is the error term.

The data set was then split into training set and test set, if size 600 and 400, respectively. Then, missing values were created in the training set under three missing

mechanisms respectively. More specifically, we made use of the function *ampute* in R package MICE to create missing data with arguments $prop = 0.3$ (the proportion of individuals who has missing data is 30%), $pattern = c(rep(1,5), rep(0,10), 1)$ (the missing pattern is: observed under the first five variables, missing under the last ten variables and the outcome is always observed). For Missing Completely at Random, we set the argument *mech* to be 'MCAR'; for Missing at Random, we set *weights* to be $c(rep(1,5), rep(0,11))$ so that missingness is only related to observed covariates; for Missing Not at Random, we set *weights* to be $rep(1,16)$ so that missingness is related to both observed and missing covariates and outcome. We create the outcome generation equation (3.2) with randomly chosen constant coefficients from 1 to 5 in such way that among the covariates from which the outcome is generated, some are missing while the others are always observed. By doing this, we want the simulation to be more generalizable in the sense that not all missing variables are related/unrelated to the outcome.

For binary outcome, we used a slightly different outcome generation equation

$$\begin{aligned} Y &= 20 + X_1 + 5e^{X_2} + X_3^4 + 5X_4^2 + X_{11}^3, \\ Y_{bi} &= I_{\{Y>35\}}. \end{aligned} \tag{3.2}$$

The constant coefficients are still randomly chosen and the way that the covariates were generated is the same as for the continuous case. The threshold is set to be 35 to mimic the structure of the real data in the next Chapter, so that the ratio of label 0 outcomes to label 1 outcomes is nearly $2:1$.

After creating missing values, we used the following methods to deal with the missing data in the random forest algorithm: multiple imputation using chain equation (4), inverse probability weighting (6)(we used the second method where the weights are used in the splitting process and prediction function), mean imputation, complete case

(ignore individuals containing missing values), the missing data method developed by [5] and mean imputation with missing indicator. 1000 trees were built in each random forest and *splitrule* was set to be *mse* for continuous outcomes and *gini* for binary outcomes. The prediction accuracy was measure by mean squared error (MSE) for continuous outcomes,

$$\text{MSE} = \frac{1}{n}(\hat{Y}_i - E[Y_i])^2,$$

and cross entropy (CE) for binary outcomes,

$$\text{CE} = -\frac{1}{n}\Big[Y_i \log \hat{Y}_i + (1 - Y_i) \log\Big(1 - \hat{Y}_i\Big)\Big].$$

Lower value of MSE/CE indicates better prediction accuracy.

3.2 Results

The prediction accuracy was calculated by either MSE or CE using six missing data techniques under three missing data mechanisms. The methods and their corresponding abbreviation were shown below.

Table 3.1 Variable description

Method abbreviation	Method
mi	multiple imputation using chain equation
ipw	inverse probability weighting
ignore	complete case
mean	mean imputation
survival	survival random forest
mean_ind	mean imputation with missing indicator

Table 3.2 Mean MSE for continuous outcome of 100 simulations

	MI	IPW	Ignore	Mean	Survival	Mean_ind
MCAR	96.142	91.630	107.543	102.675	101.117	101.193
MAR	91.547	97.819	115.960	91.601	94.003	90.410
MNAR	91.617	96.092	114.500	90.664	92.744	88.716

Table 3.3 CE for binary outcome of 100 simulations

	MI	IPW	Ignore	Mean	Survival	Mean_ind
MCAR	0.1533	0.1569	0.1632	0.1573	0.1656	0.1591
MAR	0.1532	0.1574	0.1628	0.1566	0.1666	0.1596
MNAR	0.1582	0.1578	0.1624	0.1595	0.1685	0.1631

For continuous outcome, the differences of the prediction accuracy for the six methods are quite small compared to their scales according to figures (3.1), (3.2) and (3.3). As shown by (3.2), under the three missing mechanisms, complete case is always the worst one having the largest MSE. Under MCAR, IPW exhibits smallest prediction

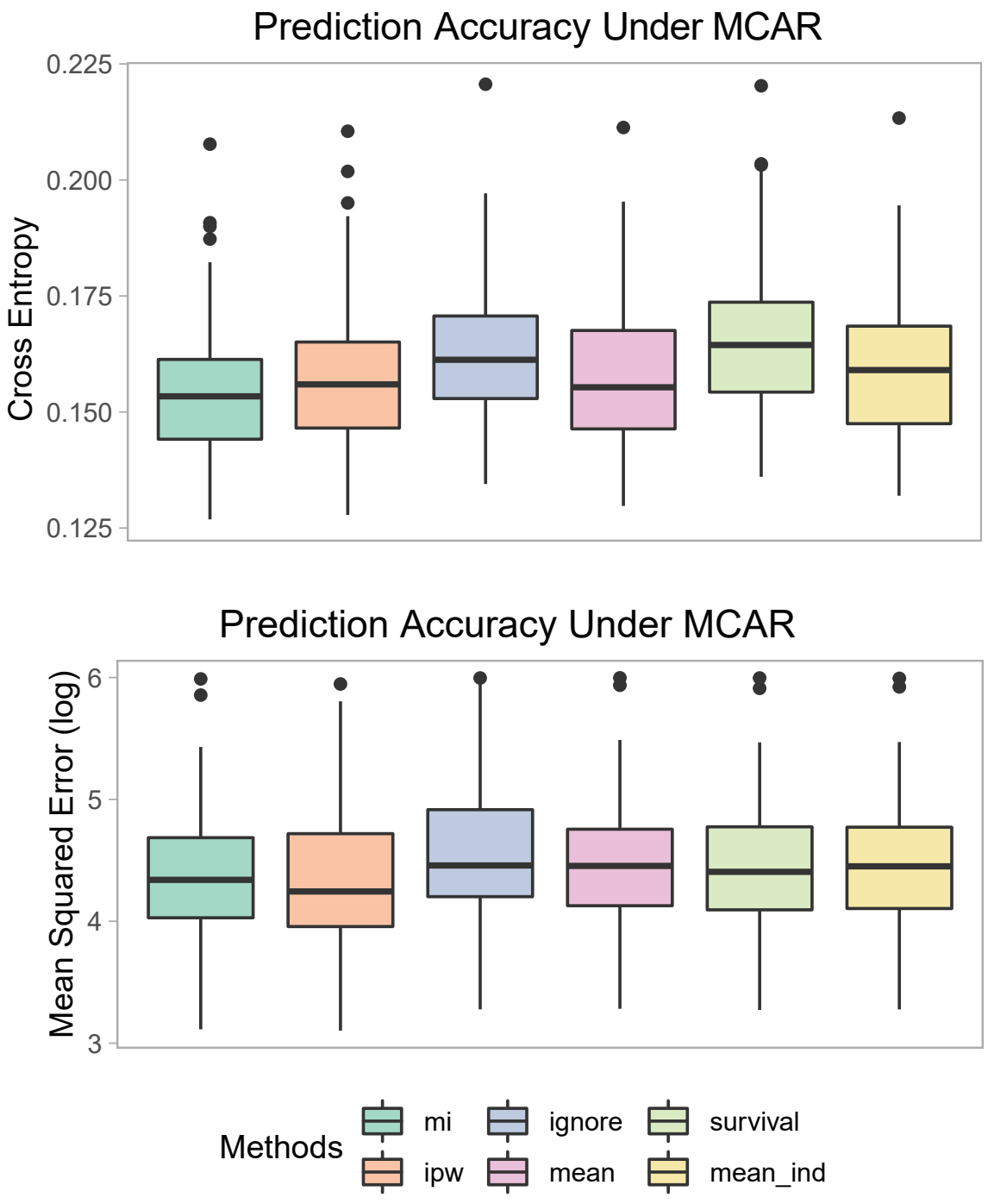

Fig. 3.1 Prediction accuracy under MCAR. The upper figure shows CE associated with binary outcome; the lower figure shows MSE associated with continuous outcome (scaled by log)

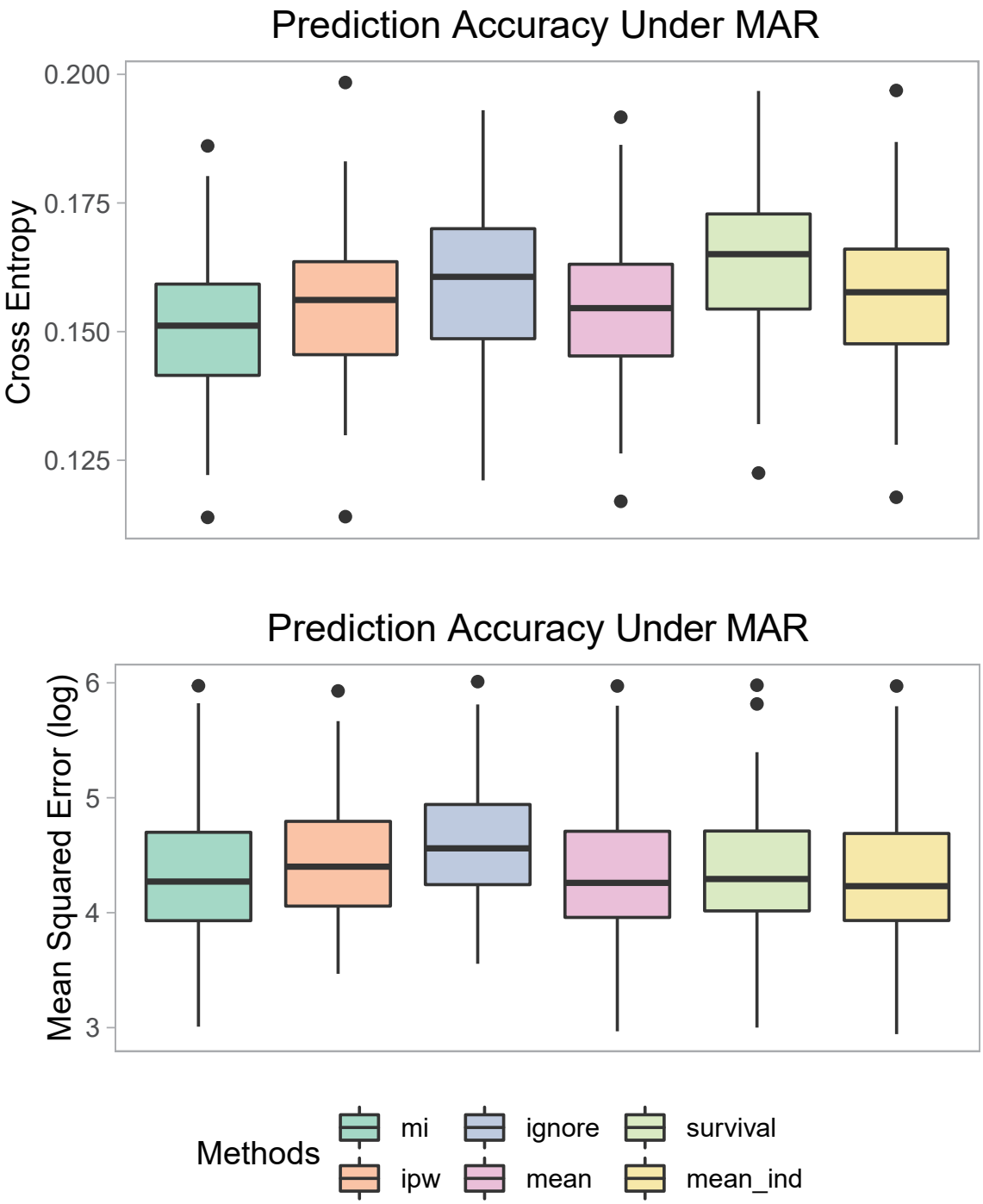

Fig. 3.2 Prediction accurary under MAR. The upper figure shows CE associated with binary outcome; the lower figure shows MSE associated with continuous outcome (scaled by log)

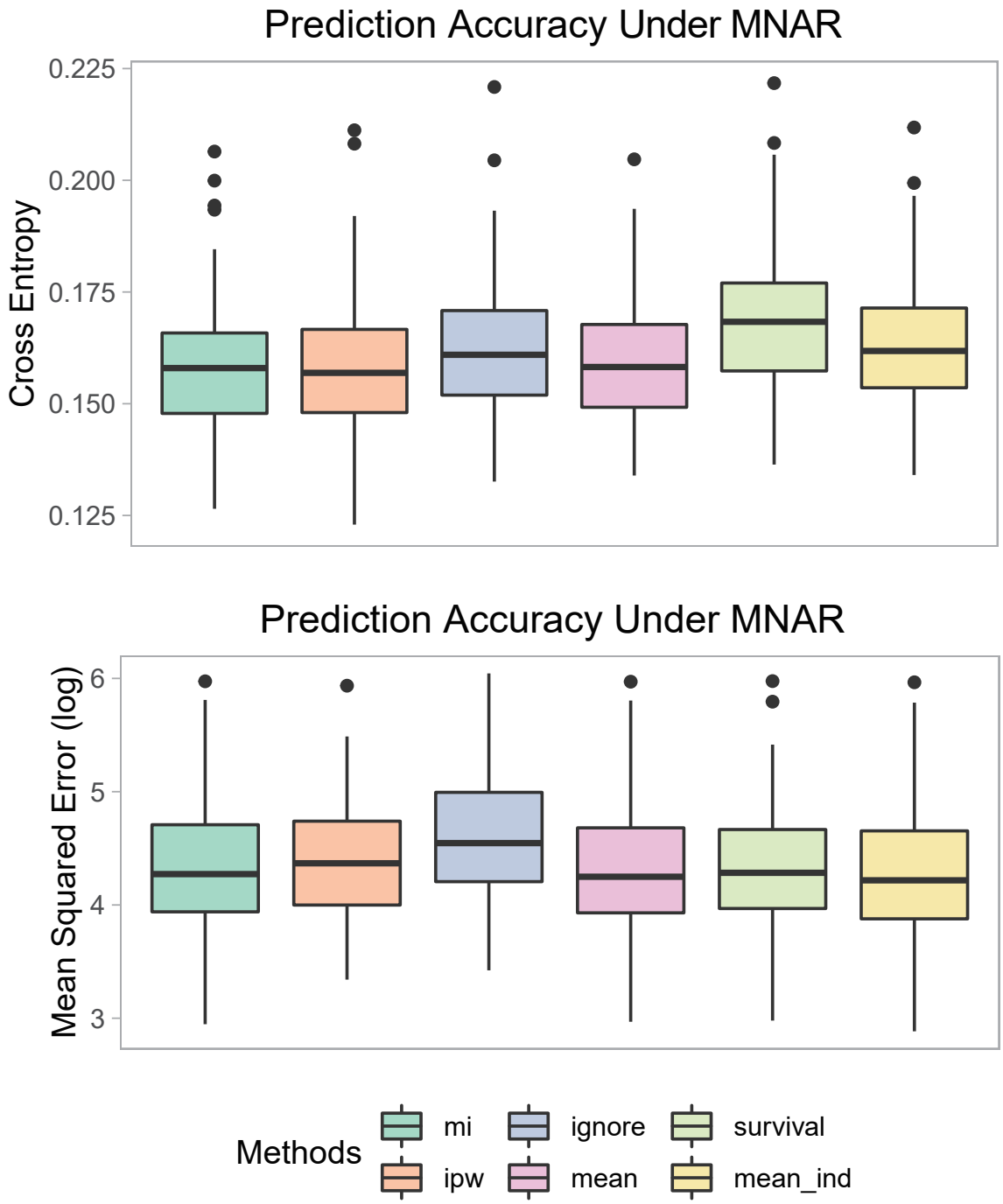

Fig. 3.3 Prediction accuracy under MNAR. The upper figure shows CE associated with binary outcome; the lower figure shows MSE associated with continuous outcome (scaled by log)

error; under MAR and MNAR, MI, mean imputation and mean imputation with indicator have lower MSE than the others, thus have the potential to yield more precise predictions. But, overall the difference between the methods is small.

For binary outcome, the differences of the prediction accuracy for the six methods are small according to the figures. However, as shown by (3.3), under the three missing mechanisms, MI, IPW and mean imputation are associated with slightly lower CE compared to complete case, survival random forest and mean imputation with indicator.

3.3 Data Analysis

3.3.1 Data Set

The data we used is the medical information of women from Pima Indian population near Phoenix, Arizona who were at the risk of diabetes [13]. This data set contains 7 covariates and 1 outcome variable:

Table 3.4 Variable description [13]

Variable Name	Variable Description
npreg	number of times pregnant
glu	plasma glucose concentration(mg/dl)
bp	diastolic blood pressure(mm Hg)
skin	triceps skin fold thickness(mm)
bmi	body mass index(kg/(m)^2)
ped	diabetes pedigree function
age	age in years
type	indicator of diabetes, Yes (1) and No (0)

The outcome *ped* is calculated as a measure of family diabetes occurence. The exact way to calculate this variable was described in [13].

This data set was split into training set and test set. The training set contains 300 observations, out of which 100 contain missing values. The missingness only happened in 3 covariates: *skin*, *bp* and *bmi*, containing 98, 3 and 13 missing values respectively. Because *bp* and *bmi* contains few missing values, we simply got rid of the observations who have missing values under these two variables to avoid imbalance. After this, we had 284 observations in the training set and 84 of them contains missing values under *skin* variable. The test set contains 332 observations with no missing values.

3.3.2 Results

Following the procedure of simulation, we implemented 5 missing data techniques with random forest on training set to build 5 models, then made predictions on test set with these models and calculated prediction accuracy, and compared these models by CE and AUC.

The results are as follows:

Table 3.5 Results of diabetes prediction

	MI	IPW	Ignore	Mean	Survival
CE	0.4819	0.4818	0.4940	0.4828	0.4766
AUC	0.8239	0.8185	0.8149	0.8246	0.8272

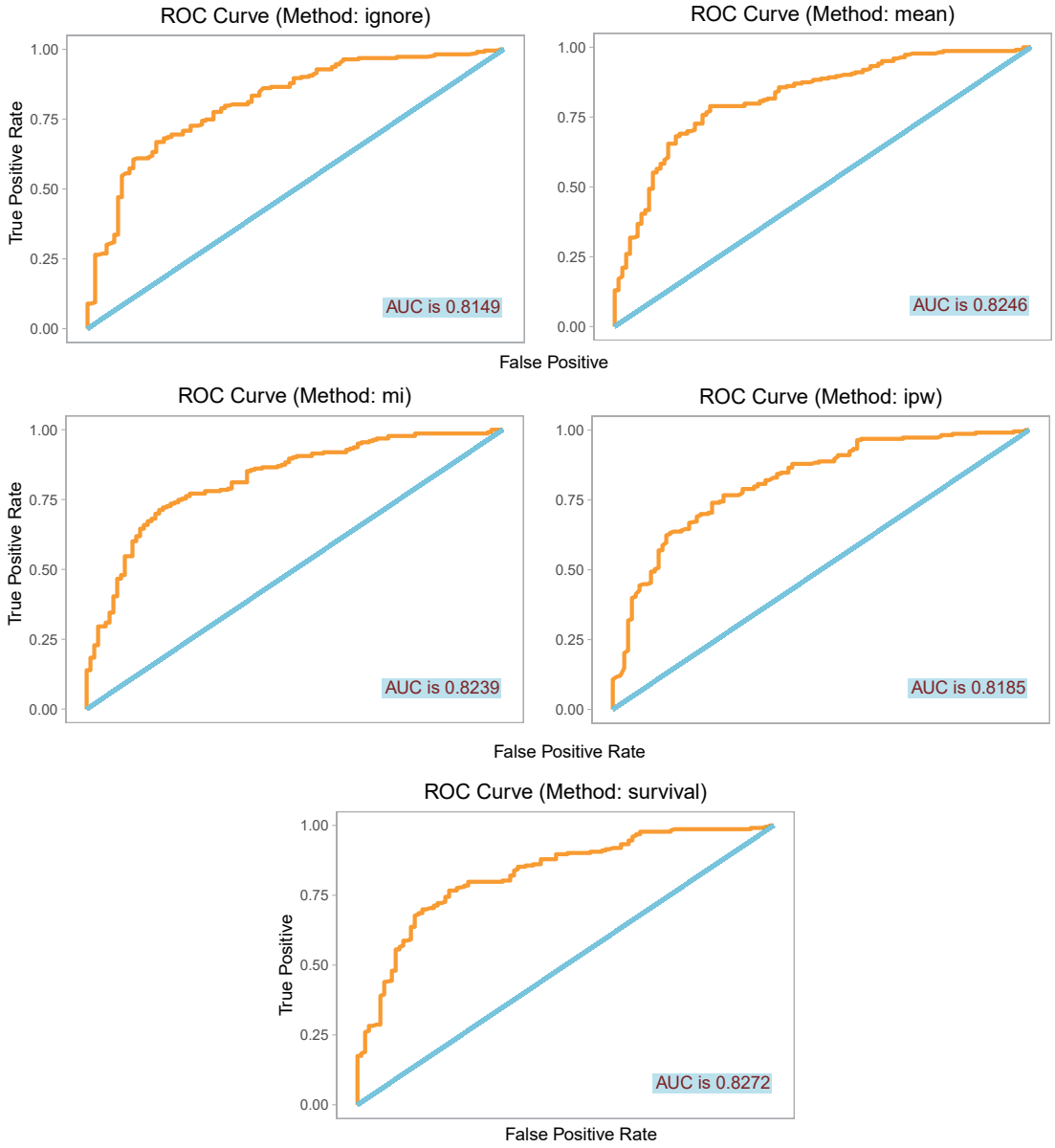

Fig. 3.4 ROC curves for the five different methods to deal with missing data

By (3.5), survival random forest is the best missing data technique in this case for it is associated with the highest AUC(0.8272) and the lowest CE(0.4766). Complete case, on the other side, is the least desirable one for it's associated with the highest CE and the lowest AUC. However, the differences of CE and AUC among these 5 methods are pretty small.

Chapter 4

Discussion

4.1 Conclusion

From the simulations, we see that except for complete case analysis, no methods can dominate the others in terms of prediction accuracy. There are many factors that can impact the performance of one method:

1. Missing mechanism: for simulation of binary outcome, MI was the best choice for both MCAR and MAR mechanisms, but in MNAR, IPW is slightly better. However, as the data missing mechanisms are assumptions that cannot be verified, in practical setting when we try to make predictions, we can try each method by splitting the data into training set and test set, and check out the prediction accuracy on the test set. Cross validation can be a better way to remove bias due to splitting.

2. Type of outcome: in the simulation part, when the outcome is continuous, survival random forest has relatively low MSE; but when outcome is binary, it is associated with the highest CE under all three missing mechanisms.

3. Missing pattern: the continuous outcome is constructed by five covariates. In the missing pattern, the first three are always observed and the last two are sometimes missing. That is, the underlying relationship is partially covered. If all the covariates

that affect the outcome are always observed and only nuisance covariates are missing, the results will be different.

4. Level of missingness: even though only missing in covariates is discussed in this project, missing in outcome can also lead to different result. For example, in binary simulation, if we allow outcome to be missing (the outcome contruction function and other missing parameters are the same), we will see that MI behaves much worse than the other methods, including complete case (4.1).

4.2 Future Work

Estimating the variability associated with the prediction estimator is an important component of predicting. [8] derived variance estimators from the random forest algorithm using subsampled random forests.

The core idea of subsampled random forest [8] is to build each tree in a certain manner so that the random forest can be viewed as random kernel U-statistic and asymptotic normal distribution can be yielded, thus variance estimator can be constructed through this distributional results.

4.2.1 Subsampled Random Forest and U-statistic

A U-statistic is defined as

$$U_n = \frac{1}{\binom{n}{k}} \sum_{(i_1,...,i_k)} h(Z_{i_1}, Z_{i_2}..., Z_{i_k}), \tag{4.1}$$

where $(i_1, ..., i_k)$ are k distinct elements from $(1, 2, ..., n)$ (which implies that $n \geqslant k$), the summation is over all $\binom{n}{k}$ combinations of $(i_1, i_2, ..., i_k)$, $h(\cdot)$ is a symmetric function with k arguments and $Z_1, Z_2, ..., Z_n$ are n independent identical random variables (or random vectors) [12].

A random forest built through algorithm (3) almost fits the structure of an incomplete U-statistic. First of all, all observed data $(\boldsymbol{X}_1, Y_i), (\boldsymbol{X}_2, Y_2), ..., (\boldsymbol{X}_n, Y_n)$ are iid (or at least assumed to be iid); secondly, when using bootstrap sample of size $k(k < n)$ to build a tree, the order of the data has no impact on the result, which corresponds to the symmetric structure of $h(\cdot)$; finally, the prediction of a random forest is the average of predictions from the $\binom{n}{k}$ trees.

However, as n grows larger, it becomes computationally infeasible to build $\binom{n}{k}$ trees. Therefore, instead of building $\binom{n}{k}$ trees, we limit the amount of trees to m_n $(m_n < \binom{n}{k})$. Also, as n increases, the prediction from each tree can be inaccurate if the number of samples k to build each tree stays the same, simply because each tree is built on insufficient information. Hence, [8] allow the number of samples k_n to increase as n increases.

The statistic that corresponds to the changes above is called resampled U-statistic [4]. It is built in similar manner as (4.1), except that instead of averaging over all the combinations of $(i_1, ..., i_k)$ sets, it averages over $m_n (m_n < \binom{n}{m})$ sets with changeable sample size k_n, and these m_n sets are selected uniformly at random with replacement from the $\binom{n}{k_n}$ combinations, that is

$$U_{n,m_n,k_n} = \frac{1}{m_n} \sum_{(i)} h_{k_n}(Z_{i_1}, ..., Z_{i_{k_n}}). \tag{4.2}$$

The only thing algorithm (3) that does not have a correspondence in (4.2) is the random selection of variables. To incorporate this, an additional randomization parameter ω was added to the statistic to incorporate the randomness brought by random selection of variables. Write

$$U_{\omega;n,k_n,m_n} = \frac{1}{m_n}\sum_{(i)} h_{k_n}^{(\omega_i)}(Z_{i_1}, Z_{i_2}, ..., Z_{i_{k_n}}), \tag{4.3}$$

which is called random kernel U-statistic. Because the variables used to build each tree are uniformly randomly selected, then all ω's can be seen as iid ($\omega_1, \omega_2, ..., \omega_{m_n} \overset{iid}{\sim} F_\omega$) and independent of the data.

Then [8] proved that such random kernel U-statistic has an asymptotic normal distribution under some mild conditions. The proof is quite long and need some techniques and theorems from other sources [6], [16], so we broken it down into two steps and state succinctly without proof.

Step 1: Let $U^*_{n,k_n,m_n} = \mathbb{E}_\omega(U_{\omega;n,k_n,m_n})$, then $U^*_{\omega;n,k_n,m_n}$ is a non-random resampled U-statistic by definition. (Just view $h^*_{k_n}(\cdot) = \mathbb{E}_\omega[h^{(\omega_i)}_{k_n}(\cdot)]$ as another symmetric function.)

It can be shown that non-random kernel U-statistic (or resampled U-statistic) has an asymptotic normal distribution. To be more specific, one of the distributional result from theorem 1 of [8] is,

$$\frac{\sqrt{n}(U^*_{n,k_n,m_n} - \theta_{k_n})}{\sqrt{k_n^2\zeta_{1,k_n}}} \xrightarrow{d} N(0,1) \tag{4.4}$$

where

$$\alpha = \lim_{n\to\infty} \frac{n}{m_n} = 0$$

$$\theta_{k_n} = \mathbb{E}h^*_{k_n}(Z_1, ..., Z_{k_n})$$

$$\zeta_{1,k_n} = cov(h^*_{k_n}(Z_1, Z_2, ..., Z_{k_n}), h^*_{k_n}(Z_1, Z'_2, ..., Z'_{k_n}).$$

Step 2: Show that

$$\sqrt{n}(U_{\omega;n,k_n,m_n} - U^*_{n,k_n,m_n}) \xrightarrow{P} 0, \tag{4.5}$$

then follows that

$$\begin{aligned}\frac{\sqrt{n}(U_{\omega;n,k_n,m_n} - \theta_{k_n})}{\sqrt{k_n^2\zeta_{1,k_n}}} &= \frac{\sqrt{n}(U_{\omega;n,k_n,m_n} - U^*_{n,k_n,m_n})}{\sqrt{k_n^2\zeta_{1,k_n}}} + \frac{\sqrt{n}(U^*_{n,k_n,m_n} - \theta_{k_n})}{\sqrt{k_n^2\zeta_{1,k_n}}} \\ &\xrightarrow{d} 0 + N(0,1) = N(0,1).\end{aligned} \tag{4.6}$$

That is, a random kernel U-statistic (corresponding to a random forest prediction) has an asymptotic normal distribution.

Having this knowledge, the variance of prediction (to be more specific, ζ_{1,k_n}) can be estimated using Monte-Carlo simulations, and 95% confidence interval can be constructed accordingly.

4.2.2 Subsampled Random Forest and Missing Data

Back to the context of missing data. In the simulation part we showed that multiple imputation and inverse probability weighting are the two best methods to handle missing data when accompanied with random forest. Multiple imputation is too complicated and algorithm-reliable to work on the theoretical properties. Thus, we are interested in how to make prediction and variance estimate under inverse probability

weighting using subsampled random forest. However, in order to do that, we need to extend the subsample random forest theory to the case inverse probability weighting.

There are some problems need to be solved before we can fit inverse probability weighting into subsampled random forest framework. First, after using inverse probability weighting, every fully observed individual is associated with a weight $w_i = \frac{1}{\hat{pi}_i}$, where $\hat{\pi}_i$ is the estimate of the true probability of being observed (1.3). In this way, the data we are about to use to build random forest $(X_1, \hat{\pi}_1, Y_1), ..., (X_m, \hat{\pi}_m, Y_m)$ are no longer iid, simply because $\hat{\pi}_i$ is estimated using all the data, and this doesn't fit in with the frame of U-statistic. A potential solution may be to prove that

$$\sqrt{m}(U_{\hat{\pi}}(Z_1, ..., Z_m) - U_{\pi}(Z_1, ..., Z_m)) \xrightarrow{P} 0. \tag{4.7}$$

Because $U_{\pi}(Z_1, ..., Z_m)$ is of the form of U-statistic, it has an asymptotic normal distribution. If 4.7 is right, then $U_{\hat{\pi}}(Z_1, ..., Z_m)$ also has an asymptotic normal distribution, and inference can be made in similar ways. We plan to explore this in future work.

References

[1] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

[2] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[3] Breiman, L. (2017). *Classification and regression trees*. Routledge.

[4] Free, E. W. (1989). Infinite order u-statistics. *Scandinavian Journal of Statistics*, 16(1):29–45.

[5] Ishwaran, H., Kogalur, U. B., Blackstone, E. H., Lauer, M. S., et al. (2008). Random survival forests. *The annals of applied statistics*, 2(3):841–860.

[6] Lee, A. J. (1990). *U- Statistics: Theory and Practice*. CRC Press.

[7] Little, R. J. and Rubin, D. B. (2014). *Statistical analysis with missing data*, volume 333. John Wiley & Sons.

[8] Mentch, L. and Hooker, G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*, 17(1):841–881.

[9] Præstgaard, J. and Wellner, J. A. (1993). Exchangeably weighted bootstraps of the general empirical process. *The Annals of Probability*, pages 2053–2086.

[10] Robins, J. M. and Rotnitzky, A. (1992). Recovery of information and adjustment for dependent censoring using surrogate markers. In *AIDS epidemiology*, pages 297–331. Springer.

[11] Rubin, D. B. (1996). Multiple imputation after 18+ years. *Journal of the American statistical Association*, 91(434):473–489.

[12] Shao, J. (2006). *Mathematical statistics: exercises and solutions*. Springer Science & Business Media.

[13] Smith, J. W., Everhart, J., Dickson, W., Knowler, W., and Johannes, R. (1988). Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 261. American Medical Informatics Association.

[14] Trevor Hastie, Robert Tibshirani, J. F. (2008). The elements of statistical learning, 2nd edition.

References

[15] Tsiatis, A. (2007). *Semiparametric theory and missing data*. Springer Science & Business Media.

[16] Van der Vaart, A. W. (2000). *Asymptotic statistics*, volume 3. Cambridge university press.

[17] White, I. R., Royston, P., and Wood, A. M. (2011). Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine*, 30(4):377–399.

Appendix

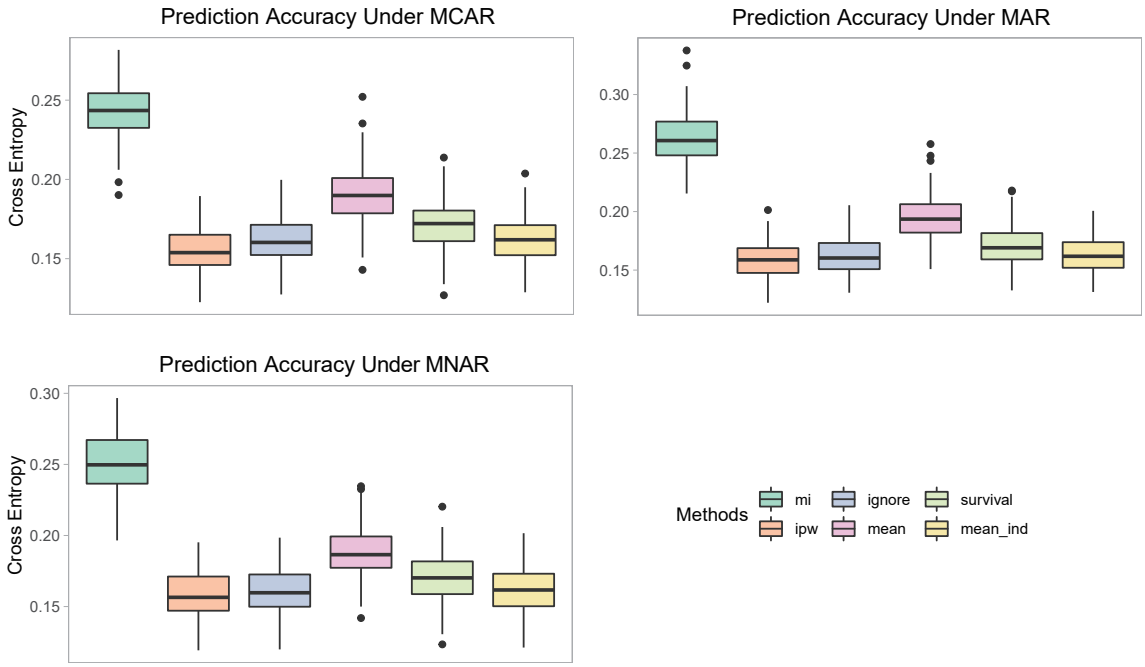

Fig. 4.1 Prediction accuracy of binary outcome when the outcome can be missing.